EDA-Laboratorio 3

30/11/2013

Mikel Barcina
Jose Ángel Gumiel

Índice de contenido

1 Introducción

En este laboratorio tenemos que trabajar de nuevo sobre el laboratorio 1. Esta vez se nos pide que diseñemos un método que, dada una lista de actores, nos permita saber si dos actores están relacionados entre sí. La relación consiste en que dos actores hayan participado en la misma película o que exista un camino a partir del cual un actor conozca a otros actores que acaban teniendo una relación con el primero de ellos. Además de saber si están relacionados tenemos que saber a cuanta distancia están el uno de otro. Si no tienen ninguna relación devuelve 0, si han participado en una misma película, son colegas, por lo que devuelve 1, si no han coincidido, pero tienen un amigo común con el que ambos han participado en distintas películas, devolverá 2, etc.

2 Diseño de las clases

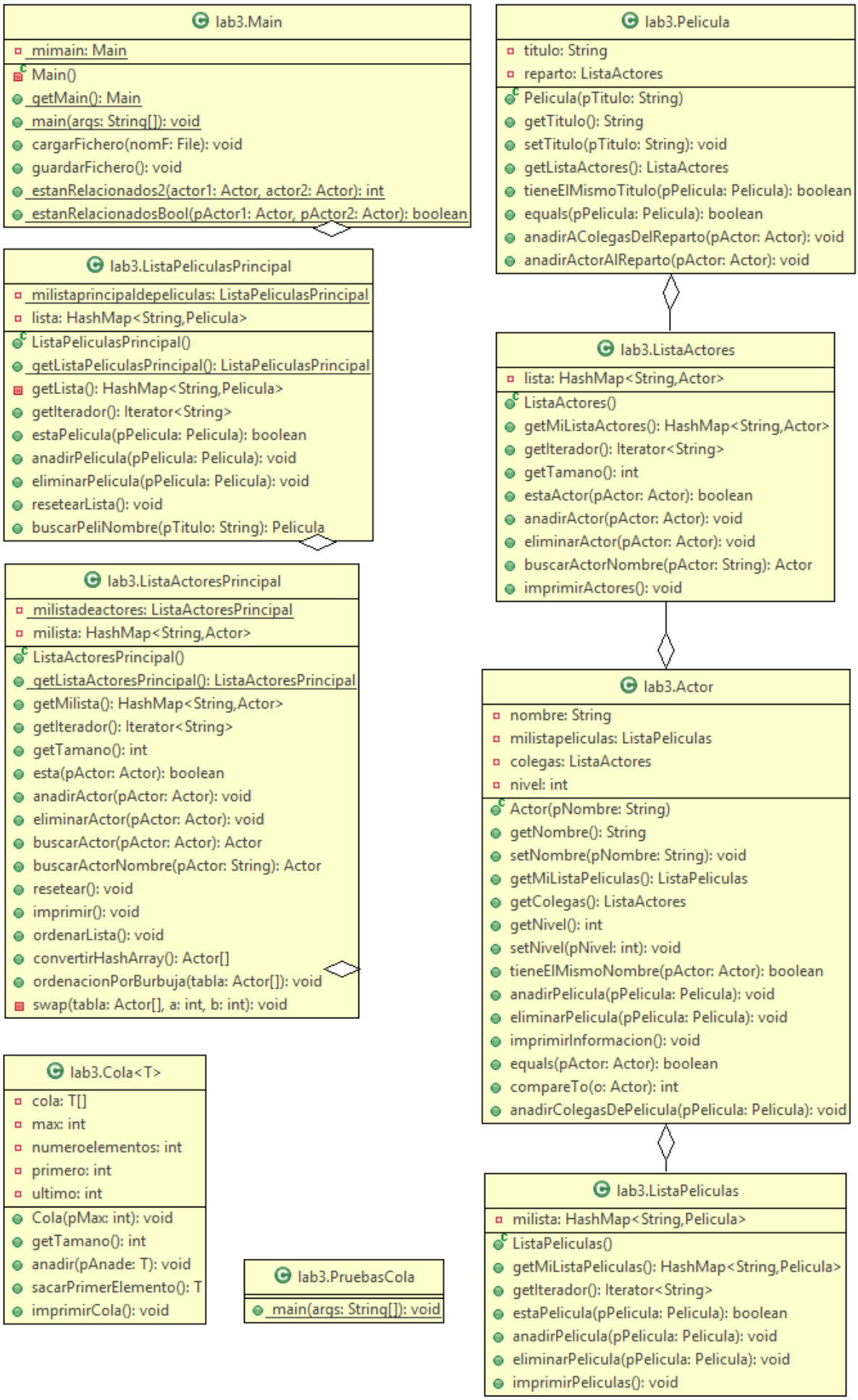
Como hemos venido haciendo hasta ahora, todas las clases que empleamos tienen los atributos privados y son accesibles mediante getters y setters. Así mismo, también serán privados los métodos que sólo vayan a ser utilizados por una única clase.

Hemos usado toda la estructura del laboratorio 1, aunque hemos corregido algunos fallos que tuvimos. Desde un primer momento diseñamos el primer laboratorio con la estructura de Hash Map, por lo que en ese aspecto no hemos tenido que cambiarlo. En este apartado expondremos solamente los cambios o incorporaciones que hayamos hecho.

En el paquete listasSimples tenemos 11 clases, de las que destacamos las siguientes:

- Main: Esta es la clase principal. Tiene el lanzador de la aplicación. Aquí es donde hemos implementado los métodos estanRelacionados() y estanRelacionadosBool(). El primero devuelve un integer, es decir, la distancia, y el segundo, solamente dice si existe un camino entre ambos actores o no, un boolean. Estos nuevos casos los hemos añadido a la interfaz de forma que sean accesibles por el usuario.
- Actor: Sobre esta clase hemos tenido que efectuar cambios. Hemos añadido un atributo de tipo ListaActores, a la que llamamos listaColegas. Esta lista contiene todos los actores con los que ha participado. También existe el atributo nivel, con sus correspondientes getters y setters, esto nos ayudará a conocer la distancia a la hora de ejecutar el método estanRelacionados(). Además también tenemos en método añadirColegas, que dada una película recorre la lista de actores (reparto) y se añaden a la lista de colegas del actor.
- Película: Aquí nos encontramos con una clase similar a la anterior. Hemos tenido que añadir un atributo de tipo ListaActores, al que llamamos reparto. Esta lista contiene a todos los actores que han participado en esa película. A su vez tenemos un método anadirActorAlReparto(), que añade un actor a la lista reparto.
- Cola: Esta clase implementa la estructura de tipo Cola. Estructura de datos que utilizamos para el método estanRelacionados, ya que tendremos una lista de elementos por examinar, en la que tenemos que coger los elementos en orden. La estructura Cola es la que mejor se adapta, ya que tomamos el primer elemento y los nuevos los insertamos al final.

En el siguiente diagrama de clases se pueden observar todas las clases mencionadas anteriormente y la totalidad de sus métodos y atributos, así como la relación existente entre ellas.


lab3.Main
mimain: Main
Main()
getMain(): Main
main(args: String[]): void
cargarFichero(nomF: File): void
guardarFichero(): void
estanRelacionados2(actor1: Actor, actor2: Actor): int
estanRelacionadosBool(pActor1: Actor, pActor2: Actor): boolean
lab3.Pelicula
titulo: String
reparto: ListaActores
Pelicula(pTitulo: String)
getTitulo(): String
setTitulo(pTitulo: String): void
getListaActores(): ListaActores
tieneElMismoTitulo(pPelicula: Pelicula): boolean
equals(pPelicula: Pelicula): boolean
anadirAColegasDelReparto(pActor: Actor): void
anadirActorAlReparto(pActor: Actor): void
lab3.ListaPeliculasPrincipal
milistaprincipaldepeliculas: ListaPeliculasPrincipal
lista: HashMap<String,Pelicula>
ListaPeliculasPrincipal()
getListaPeliculasPrincipal(): ListaPeliculasPrincipal
getLista(): HashMap<String,Pelicula>
getIterador(): Iterator<String>
estaPelicula(pPelicula: Pelicula): boolean
anadirPelicula(pPelicula: Pelicula): void
eliminarPelicula(pPelicula: Pelicula): void
resetearLista(): void
buscarPeliNombre(pTitulo: String): Pelicula
lab3.ListaActores
lista: HashMap<String,Actor>
ListaActores()
getMiListaActores(): HashMap<String,Actor>
getIterador(): Iterator<String>
getTamano(): int
estaActor(pActor: Actor): boolean
anadirActor(pActor: Actor): void
eliminarActor(pActor: Actor): void
buscarActorNombre(pActor: String): Actor
imprimirActores(): void
lab3.ListaActoresPrincipal
milistadeactores: ListaActoresPrincipal
milista: HashMap<String,Actor>
ListaActoresPrincipal()
getListaActoresPrincipal(): ListaActoresPrincipal
getMilista(): HashMap<String,Actor>
getIterador(): Iterator<String>
getTamano(): int
esta(pActor: Actor): boolean
anadirActor(pActor: Actor): void
eliminarActor(pActor: Actor): void
buscarActor(pActor: Actor): Actor
buscarActorNombre(pActor: String): Actor
resetear(): void
imprimir(): void
ordenarLista(): void
convertirHashArray(): Actor[]
ordenacionPorBurbuja(tabla: Actor[]): void
swap(tabla: Actor[], a: int, b: int): void
lab3.Actor
nombre: String
milistapeliculas: ListaPeliculas
colegas: ListaActores
nivel: int
Actor(pNombre: String)
getNombre(): String
setNombre(pNombre: String): void
getMiListaPeliculas(): ListaPeliculas
getColegas(): ListaActores
getNivel(): int
setNivel(pNivel: int): void
tieneElMismoNombre(pActor: Actor): boolean
anadirPelicula(pPelicula: Pelicula): void
eliminarPelicula(pPelicula: Pelicula): void
imprimirInformacion(): void
equals(pActor: Actor): boolean
compareTo(o: Actor): int
anadirColegasDePelicula(pPelicula: Pelicula): void
lab3.Cola<T>
cola: T[]
max: int
numeroelementos: int
primero: int
ultimo: int
Cola(pMax: int): void
getTamano(): int
anadir(pAnade: T): void
sacarPrimerElemento(): T
imprimirCola(): void
lab3.PruebasCola
main(args: String[]): void
lab3.ListaPeliculas
milista: HashMap<String,Pelicula>
ListaPeliculas()
getMiListaPeliculas(): HashMap<String,Pelicula>
getIterador(): Iterator<String>
estaPelicula(pPelicula: Pelicula): boolean
anadirPelicula(pPelicula: Pelicula): void
eliminarPelicula(pPelicula: Pelicula): void
imprimirPeliculas(): void

3 Descripción de las estructuras de datos principales

Para este laboratorio hemos usado dos estructuras de datos diferentes, Hash Map y Cola.

- HashMap: Es una estructura de datos claves (keys) con valores (values). La motivación para usar esta estructura es que ofrece un soporte muy eficiente para la búsqueda de datos. Se puede acceder a los elementos de la tabla con una clave generada. El funcionamiento del HashMap consiste en transformar la clave en un hash, un número que identifica la posición en la que se encuentra el valor deseado.

  Esta estructura gana utilidad cuando trabajamos con cantidades de información considerables, tal y como ocurre en nuestro caso.

- Cola: La particularidad de una estructura de datos de cola es el hecho de que sólo podemos acceder al primer y al último elemento de la estructura. Así mismo, los elementos sólo se pueden eliminar por el principio y sólo se pueden añadir por el final de la cola. Nos ha resultado especialmente útil, como hemos mencionado anteriormente, para tomar los actores por examinar.

4 Diseño e implementación de los métodos principales

En este apartado consideramos que los métodos principales son dos, el que mide la distancia y el que nos dice si están relacionados, es por ello que sólo nos centraremos en esos.

Clase Main:

- Método estanRelacionados(). Dados dos actores nos dice su relación entre ellos mediante un entero. Para este método utilizamos dos estructuras diferentes, una Cola para los elementos que aún están por examinar, y una lista de tipo Hash Map para los actores que ya están examinados.

  En un primer lugar seteamos los niveles a 1. A continuación añadimos el actor que queremos comparar a la cola porExaminar. Empieza el primer while, sólo se saldrá de él si el booleano enc es true o si la cola está vacía.

  Se compara si el primer elemento de la cola está en la lista de colegas del actor2. Si es así devuelve true y devolvemos el nivel. De lo contrario, cogemos los colegas del actor por examinar y los añadimos a la cola cambiando su nivel al nivel del actor anterior pero sumándole uno.

  Dado que si volvemos a ejecutar el programa de nuevo obtendríamos resultados incorrectos porque hemos hecho modificaciones en el atributo nivel de los actores, lo que hacemos es guardar la distancia en una variable para proceder a restaurar los actores a su estado original. En el caso de la cola cogemos el primer elemento, lo buscamos en la ListaActoresPrincipal y ponemos su nivel a cero. Con la lista de examinados hacemos lo mismo, pero como es un Hash Map tenemos que utilizar un iterador. Al final se quedan las listas vacías y los actores tal y como estaban.

- Método estanRelacionadosBool(). Como el primer método que diseñamos fue el que devolvía la distancia, este método que devuelve un valor booleano es muy sencillo. Ejecuta el método expuesto anteriormente, y si el valor es mayor que 0 devuelve True, afirmando que sí que existe una relación entre los dos actores.

Añadir que en la especificación no decía qué es lo que tenía que devolver si comparamos la relación entre un actor y él mismo de nuevo. Lo incluimos en el caso general.

4.1 Alternativas examinadas

La primera parte del laboratorio fue sencilla y apenas nos planteó dudas. Sabíamos que necesitaríamos una lista de colegas en la clase Actor y una lista de actores en la clase Película. Esto no tuvo ninguna complicación y lo hicimos bastante rápido.

En nuestro caso no fuimos a hacer si estaban relacionadas o no mediante un booleano, fuimos a por el método más complejo que era el que además tenía que devolver distancia. Fue un método que aunque lo viéramos bastante claro desde un punto de vista ajeno a la codificación, posteriormente nos dio unos cuantos quebraderos de cabeza.

Al principio estábamos añadiendo casos no generales para descartar y utilizábamos una variable de tipo distancia, la cual no nos funcionaba correctamente y generaba resultados incorrectos. Para este método hemos desarrollado por lo menos cuatro métodos diferentes, dos de ellos han funcionado, pero al final nos quedamos con el más corto y sencillo de los dos. Nuestras soluciones empezaron a salir cuando decidimos poner en la clase Actor un atributo llamado nivel.

5 Código

Hemos intentado poner el código de una forma que se adapte a la página, sin embargo, hay líneas de código que son excesivamente largas y nos es imposible adaptarlas mejor a este documento.

5.1 Programa

A continuación mostramos la totalidad del código que hemos implementado, ordenado por clases:

5.1.1 Clase Main

```
package lab3;

import java.io.*;  //Importo el Paquete Entero
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class Main {
	//Atributos
	private static Main mimain = new Main();


	//Constructora
	private Main(){	}


	//Getters y Setters
	public static Main getMain(){
		return mimain;
	}


	//Otros Métodos

	public static void main(String[] args){
		boolean repetir;
		repetir = true;
		JOptionPane.showMessageDialog(null, "Bienvenido a la base virtual de cine");
		do{
			String entrada = JOptionPane.showInputDialog("¿Qué quieres hacer?\n"
					+ "1- Cargar fichero\n"
					+ "2- Añadir un actor\n"
```

```
                                        + "3- Añadir una pelicula\n"
                                        + "4- Ordenar la lista de actores\n"
                                        + "5- Guardar fichero\n"
                                        + "6- Salir\n"
                                        + "7- Imprimir colegas de un actor\n"
                                        + "8- Ver relación entre dos actores (distancia)\n"
                                        + "9- ¿Estan relacionados los actores? (boolean)");
                        switch(entrada){
                        case "1":
                                JFileChooser fc = new JFileChooser();
                                fc.setCurrentDirectory(new File("."));
                                fc.setDialogTitle("Elige un fichero");
                                fc.setAcceptAllFileFilterUsed(false);
                                if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) { //Comprueba que hemos seleccionado un archivo
                                        Main.getMain().cargarFichero(fc.getSelectedFile());
                                } else {
                                        System.out.println("No seleccion ");
                                }
                                break;
                        case "2":
                                Actor anadir = new Actor(JOptionPane.showInputDialog("Introduce el nombre del actor"));
                                Pelicula anadepelicula;
                                if(ListaActoresPrincipal.getListaActoresPrincipal().esta(anadir)){
                                        System.out.println("El actor ya se encuentra en la lista"); //ListaActores lo comprueba también pero poniendo esto aqui te ahorras meterle peliculas para nada.
                                }else{
                                        boolean quedanpeliculas = true;
                                        String respuesta;
                                        do{
                                                respuesta = JOptionPane.showInputDialog("¿Quieres añadirle una película? (Si/No)");
                                                if (respuesta.equalsIgnoreCase("si")){
                                                        anadepelicula = new Pelicula(JOptionPane.showInputDialog("Introduce el titulo de la película"));
                                                        anadir.anadirPelicula(anadepelicula);
                                                }else if (respuesta.equalsIgnoreCase("no")){
                                                        quedanpeliculas=false;
                                                }else{
                                                        JOptionPane.showMessageDialog(null, "La respuesta introducida es incorrecta. Por favor introduzca si o no");
                                                }
                                        }while (quedanpeliculas);
                                        ListaActoresPrincipal.getListaActoresPrincipal().anadirActor(anadir);
                                }
                                break;
                        case "3":
                                ListaPeliculasPrincipal.getListaPeliculasPrincipal().anadirPelicula(new Pelicula(JOptionPane.showInputDialog("Introduce el titulo de la película")));
                                break;
                        case "4":
                                ListaActoresPrincipal.getListaActoresPrincipal().ordenarLista();
                                break;
                        case "5":
                                Main.getMain().guardarFichero();
                                break;
                        case "6":
                                repetir=false;
                                break;
                        case "7":
                                String auuuux = JOptionPane.showInputDialog("Introduce el nombre del actor");
                                Actor aaux = ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(auuuux);
                                if(aaux!=null){
                                        aaux.imprimirInformacion();
                                }else{
                                        JOptionPane.showMessageDialog(null, "El actor introducido es incorrecto");
                                }
                                break;
                        case "8":
                                String compara1 = JOptionPane.showInputDialog("Introduce el nombre del actor");
                                String compara2 = JOptionPane.showInputDialog("Introduce el nombre del actor");
                                Actor acompara1 = ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(compara1);
                                Actor acompara2 = ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(compara2);
                                int distancia = 0;
                                if((acompara1!=null)&&(acompara2!=null)){
                                        distancia = estanRelacionados(acompara1,acompara2);

                                }else{
                                        JOptionPane.showMessageDialog(null,"Uno de los actores introducidos no se encuentra en la Lista de Actores");
                                }
                                System.out.println(distancia);
                                break;
                        case "9":
                                String compar1 = JOptionPane.showInputDialog("Introduce el nombre del actor");
                                String compar2 = JOptionPane.showInputDialog("Introduce el nombre del actor");
```

```
                                        Actor acompar1 = ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(com-
par1);
                                        Actor acompar2 = ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(com-
par2);
                                        boolean relacionados = false;
                                        if((acompar1!=null)&&(acompar2!=null)){
                                                relacionados = estanRelacionadosBool(acompar1,acompar2);

                                        }else{
                                                JOptionPane.showMessageDialog(null,"Uno de los actores introducidos no se en-
cuentra en la Lista de Actores");
                                        }
                                        System.out.println(relacionados);
                                        break;
                                default:
                                        JOptionPane.showMessageDialog(null, "La opción introducida es incorrecta. Introduce un
número del 1 al 5.");
                                }
                        }while(repetir==true);
                }



                public void cargarFichero(File nomF){
                        try{
                                FileReader fr = new FileReader(nomF);
                                @SuppressWarnings("resource")
                                BufferedReader br = new BufferedReader(fr);
                                String nombreactor;
                                String linea, tituloaux;
                                Actor ultimoactor = null;
                                Pelicula anadepelicula;
                                boolean todobien = true;
                                int ayuda = 0;
                                while ((linea=br.readLine())!=null) {
                                        ayuda=0;

                                        do{
                                                try{
                                                        todobien=true;
                                                        if(linea.length()!=0){
                                                                if(linea.substring(0,3).equals("\t\t\t")){
                                                                        if(ultimoactor!=null){
                                                                                String[] sintabuladores = li-
nea.split("\t");

                                                                                String[] titulo = sintabuladores[3-
ayuda].split(" *[(]+\\d+[)]");

                                                                                tituloaux = titulo[0].repla-
ceAll("[\"]","");



                                                                                //Tenemos el título de la película
                                                                                if(tituloaux.equals("")){
                                                                                        todobien=false;
                                                                                        ayuda++;
                                                                                }else{
                                                                                        anadepelicula = ListaPeliculas-
Principal.getListaPeliculasPrincipal().buscarPeliNombre(tituloaux);

                                                                                        if(anadepelicula==null){
                                                                                                //No está en la Lista
Principal de Peliculas

                                                                                                anadepelicula = new
Pelicula(tituloaux);

                                                                                                ListaPeliculasPrinci-
pal.getListaPeliculasPrincipal().anadirPelicula(anadepelicula);

                                                                                                ultimoactor.anadirPe-
licula(anadepelicula);

                                                                                        }else{
                                                                                                ultimoactor.anadirPe-
licula(anadepelicula);

                                                                                        }
                                                                                }



                                                                        }else{
                                                                                JOptionPane.showMessageDialog(null, "Ha
ocurrido un error");

                                                                        }
                                                                }else{
                                                                        String[] division = linea.split("\t");
                                                                        nombreactor=division[0];


                                                                        //Tenemos el nombre del actor
                                                                        if(ayuda==0){
                                                                                ultimoactor = ListaActoresPrinci-
pal.getListaActoresPrincipal().buscarActorNombre(nombreactor);

                                                                                if (ultimoactor==null){
```

```
                                                                        //No está en la Lista Principal
de Actores
                                                                        ultimoactor = new Actor(nombre-
actor);
                                                                        ListaActoresPrincipal.getLis-
taActoresPrincipal().anadirActor(ultimoactor);
                                                                }
                                                        }


*[(]+\\d+[)]");
                                                        String[] titulo = division[2-ayuda].split("

                                                        tituloaux = titulo[0].replaceAll("\"","");


                                                       //Tenemos el titulo de la película
                                                        if(tituloaux.equals("\t")){
                                                                todobien=false;
                                                                ayuda++;
                                                        }else{
                                                                anadepelicula = ListaPeliculasPrinci-
pal.getListaPeliculasPrincipal().buscarPeliNombre(tituloaux);
                                                                if(anadepelicula==null){
                                                                        anadepelicula = new Pelicula(ti-
tuloaux);
                                                                        ListaPeliculasPrincipal.getLis-
taPeliculasPrincipal().anadirPelicula(anadepelicula);
                                                                        ultimoactor.anadirPelicula(ana-
depelicula);
                                                                }else{
                                                                        ultimoactor.anadirPelicula(ana-
depelicula);
                                                                }
                                                        }
                                                 }
                                        }
                                }catch(ArrayIndexOutOfBoundsException ae){
                                                        //A veces en vez de salir actriz\t\tpelicula solo hay un \t.
Lo solucionamos aqui:
                                                        todobien=false;
                                                        ayuda++;
                                                        ListaActoresPrincipal.getListaActoresPrincipal().eliminarAc-
tor(ultimoactor); //Para evitar duplicados
                                                }
                                        }while(!todobien);
                                }
                        }catch(IOException e) {
                                e.printStackTrace();
                        }
         }


          public void guardarFichero(){
                    FileWriter fichero = null;
        PrintWriter pw = null;
        try
        {
            fichero = new FileWriter("C:\\Documents and Settings\\euitibi\\Escritorio\\actrices.txt");
            pw = new PrintWriter(fichero);
            String auxAct, auxPel;

            Iterator<String> itrAct = ListaActoresPrincipal.getListaActoresPrincipal().getIterador();
            Iterator<String> itrPel;

            while(itrAct.hasNext()){
                  auxAct = itrAct.next();
                  pw.println(auxAct);
                  itrPel = ListaActoresPrincipal.getListaActoresPrincipal().getMilista().get(auxAct).getMiListaPelicu-
las().getIterador();
                  while(itrPel.hasNext()){
                            auxPel=itrPel.next();
                            pw.println("--> "+auxPel);
                  }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

        finally {
           try {
                    if (null != fichero)
                       fichero.close();
           } catch (Exception e2) {
              e2.printStackTrace();
           }
        }
         }
```

```
	public static int estanRelacionados(Actor actor1, Actor actor2){
		boolean enc = false;
		Cola<Actor> porExaminar = new Cola(ListaActoresPrincipal.getListaActoresPrincipal().getTamano());
		ListaActores examinados = new ListaActores();
		porExaminar.anadir(actor1);
		Actor actoraexaminar = new Actor("Actor por Defecto");
		Iterator<String> iteradordecolegas;
		String saux;
		Actor aaux;
		int nivel = 1;
		actor1.setNivel(nivel);//Porque si lo encuentra de primeras devuelve 0

		while((enc==false)&&(porExaminar.getTamano()>0)){
			actoraexaminar = porExaminar.sacarPrimerElemento();
			if(actoraexaminar.getColegas().estaActor(actor2)){
				enc=true;
			}else{
				iteradordecolegas = actoraexaminar.getColegas().getIterador();
				nivel=actoraexaminar.getNivel()+1;
				while(iteradordecolegas.hasNext()){
					saux = iteradordecolegas.next();
					aaux = actoraexaminar.getColegas().getMiListaActores().get(saux);
					if((aaux!=null)&&(!examinados.estaActor(aaux))){
						aaux.setNivel(nivel);
						porExaminar.anadir(aaux);
						examinados.anadirActor(aaux);
					}
				}
			}
		}
		int resultado=actoraexaminar.getNivel();
		while(porExaminar.getTamano()!=0){
			Actor a=porExaminar.sacarPrimerElemento();
			ListaActoresPrincipal.getListaActoresPrincipal().buscarActor(a).setNivel(0);
			//Esto borra la lista de actores porExaminar y ademas restaura su valor a 0.
		}
		while(examinados.getIterador().hasNext()){
			String a1=examinados.getIterador().next();
			ListaActoresPrincipal.getListaActoresPrincipal().buscarActorNombre(a1).setNivel(0);
			examinados.eliminarActor(examinados.buscarActorNombre(a1));
			/*Hace lo mismo con los examinados. Esto es porque si lo dejamos igual y
			volvemos a hacer otra búsqueda, el resultado no sería correcto, ya que
			tendrían niveles asignados.*/
		}

		return resultado;
	}

	public static boolean estanRelacionadosBool(Actor pActor1, Actor pActor2){
		boolean enc=false;
		if(estanRelacionados(pActor1, pActor2)>0){
			enc=true;
		}
		return enc;
	}
}
```

5.1.2 Clase Actor

```
package lab3;

import java.util.Iterator;

public class Actor{
	//Atributos
	private String nombre;
	private ListaPeliculas milistapeliculas;
	private ListaActores colegas;
	private int nivel; //Mide distancia en el estan relacionados

	//Constructora
	public Actor(String pNombre){
		this.nombre=pNombre;
		this.milistapeliculas = new ListaPeliculas();
		this.colegas=new ListaActores();
		this.nivel = 0;
	}
```

```java
//Getters y Setters
public String getNombre() {
	return nombre;
}

public void setNombre(String pNombre) {
	this.nombre = pNombre;
}

public ListaPeliculas getMiListaPeliculas(){
	return this.milistapeliculas;
}

public ListaActores getColegas(){
	return this.colegas;
}

public int getNivel(){
	return this.nivel;
}

public void setNivel(int pNivel){
	this.nivel=pNivel;
}

//Otros Metodos
public boolean tieneElMismoNombre(Actor pActor){
	try{
		if(pActor.getNombre().equals(this.getNombre())){
			return true;
		}else
			return false;
	}
	catch (Exception e){
		return false;
	}
}

public void anadirPelicula(Pelicula pPelicula){
	try{
		 if(pPelicula.getListaActores().getTamano()==0){
			this.getMiListaPeliculas().anadirPelicula(pPelicula);
			pPelicula.getListaActores().anadirActor(this);
		 }else{
			pPelicula.anadirAColegasDelReparto(this);
			this.anadirColegasDePelicula(pPelicula);
			pPelicula.anadirActorAlReparto(this);
			this.getMiListaPeliculas().anadirPelicula(pPelicula);
		 }
	}catch(Exception e){
		System.out.println("La pelicula introducida no es válida");
	}
}

public void eliminarPelicula(Pelicula pPelicula){
	this.getMiListaPeliculas().eliminarPelicula(pPelicula);
}

public void imprimirInformacion(){
	System.out.println("Nombre: "+this.getNombre());
	System.out.println("Peliculas:");
	this.getMiListaPeliculas().imprimirPeliculas();
	System.out.println("Colegas:");
	this.colegas.imprimirActores();
}
```

```
	public boolean equals(Actor pActor){
		return this.tieneElMismoNombre(pActor);
	}


	public int compareTo(Actor o) {
		return this.getNombre().compareTo(o.getNombre());
	}

	public void anadirColegasDePelicula(Pelicula pPelicula){
		Iterator <String> it=pPelicula.getListaActores().getIterador();
		while(it.hasNext()){
			String aux;
			Actor aaux;
			aux=it.next();
			if (aux!=null){
				aaux=pPelicula.getListaActores().buscarActorNombre(aux);
				if ((aaux!=null)&&(!aux.equals(this.getNombre()))){
					this.getColegas().anadirActor(aaux);
				}
			}
		}
	}
}
```

5.1.3 Clase Película

```
package lab3;

import java.util.Iterator;

public class Pelicula {
	//Atributos
	private String titulo;
	private ListaActores reparto;

	//Constructora
	public Pelicula(String pTitulo){
		this.titulo=pTitulo;
		this.reparto=new ListaActores();
	}


	//Getters y Setters
	public String getTitulo(){
		return this.titulo;
	}

	public void setTitulo(String pTitulo){
		this.titulo=pTitulo;
	}

	public ListaActores getListaActores(){
		return this.reparto;
	}


	//Otros Metodos
	public boolean tieneElMismoTitulo(Pelicula pPelicula){
		try{
			if(pPelicula.getTitulo().equals(this.getTitulo())){
				return true;
			}else
				return false;
		}
		catch (Exception e){
			return false;
		}
```

```
	}

	public boolean equals(Pelicula pPelicula){
		return this.tieneElMismoTitulo(pPelicula);
	}

	public void anadirAColegasDelReparto(Actor pActor){
		try{
			Iterator<String> it = reparto.getIterador();
			while(it.hasNext()){
				String saux = it.next();
				if(!saux.equals(pActor.getNombre())){
					Actor aaux = reparto.getMiListaActores().get(saux);
					aaux.getColegas().anadirActor(pActor);
				}
			}
		}catch(Exception e){
			System.out.println("El actor introducido no es válido");
		}
	}

	public void anadirActorAlReparto(Actor pActor){
		if(!this.getListaActores().getMiListaActores().containsKey(pActor.getNombre())){
			this.getListaActores().anadirActor(pActor);
		}
	}
}
```

5.1.4 Clase ListaActores

```
package lab3;

import java.util.HashMap;
import java.util.Iterator;

public class ListaActores {

	private HashMap<String, Actor> lista;

	public ListaActores(){
		this.lista = new HashMap<String, Actor>();
	}

	//Getters y Setters
	public HashMap<String,Actor> getMiListaActores(){
		return lista;
	}

	public Iterator<String> getIterador(){
		return this.getMiListaActores().keySet().iterator();
	}

	public int getTamano(){
		return this.lista.size();
	}


	//Otros Métodos
	public boolean estaActor(Actor pActor){
		try{
			return this.getMiListaActores().containsKey(pActor.getNombre());
		}catch(NullPointerException e){
			System.out.println("El Actor que estás buscando no se encuntra en la lista");
			return false;
		}
	}

	public void anadirActor(Actor pActor){
		 if(pActor!=null){
			if(!this.estaActor(pActor)){
```

```
                                if(!ListaActoresPrincipal.getListaActoresPrincipal().esta(pActor)){
                                        ListaActoresPrincipal.getListaActoresPrincipal().anadirActor(pActor);
                                        this.getMiListaActores().put(pActor.getNombre(),pActor);
                                }else{
                                        this.getMiListaActores().put(pActor.getNombre(),pActor);
                                }
                        }
                }
        }

        public void eliminarActor(Actor pActor){
                if(pActor!=null){
                        if(estaActor(pActor)){
                                this.getMiListaActores().remove(pActor.getNombre());
                        }else{
                                System.out.println("La película no se encuentra en la lista");
                        }
                }
        }

        public Actor buscarActorNombre(String pActor){
                //Devuelve el Actor si está y sino devuelve null
                return this.getMiListaActores().get(pActor);
        }

        public void imprimirActores(){
                Iterator<String> it = this.getIterador();
                while(it.hasNext()){
                        System.out.println("->"+it.next());
                }
        }
}
```

5.1.5 Clase ListaPeliculas

```
package lab3;

import java.util.HashMap;
import java.util.Iterator;

public class ListaPeliculas {
        //Atributos
        private HashMap<String, Pelicula> milista;

        //Constructora
        public ListaPeliculas(){
                this.milista = new HashMap<String, Pelicula>();
        }


        //Getter y Setters
        public HashMap<String, Pelicula> getMiListaPeliculas(){
                return this.milista;
        }

        public Iterator<String> getIterador(){
                return this.getMiListaPeliculas().keySet().iterator();  //Esto crea un iterador de keys o
llaves

//en nuestro caso titulos de peliculas.
        }


        //Otros Metodos
        public boolean estaPelicula(Pelicula pPelicula){
                try{
                        String pTitulo = pPelicula.getTitulo();
                        return this.getMiListaPeliculas().containsKey(pTitulo);
                }catch(NullPointerException e){
                        return false;
                }
        }

        public void anadirPelicula(Pelicula pPelicula){
                 if(pPelicula!=null){
                        if(!this.estaPelicula(pPelicula)){
```

```
                                if(!ListaPeliculasPrincipal.getListaPeliculasPrincipal().estaPelicula(pPeli-
cula)){
                                        ListaPeliculasPrincipal.getListaPeliculasPrincipal().anadirPeli-
cula(pPelicula);
                                        this.getMiListaPeliculas().put(pPelicula.getTitulo(),pPelicula);
                                }else{
                                        this.getMiListaPeliculas().put(pPelicula.getTitulo(),pPelicula);
                                }
                        }
                }
        }

        public void eliminarPelicula(Pelicula pPelicula){
                if(pPelicula!=null){
                        if(estaPelicula(pPelicula)){
                                this.getMiListaPeliculas().remove(pPelicula.getTitulo());
                        }else{
                                System.out.println("La película no se encuentra en la lista");
                        }
                }
        }

        public void imprimirPeliculas(){
                Iterator<String> it = this.getIterador();
                while(it.hasNext()){
                        System.out.println("->"+it.next());
                }
        }

}
```

5.1.6 Clase ListaActoresPrincipal

```
package lab3;

import java.util.HashMap;
import java.util.Iterator;

//import javax.swing.JOptionPane;


public class ListaActoresPrincipal {
        //Atributos
        private static ListaActoresPrincipal milistadeactores = new ListaActoresPrincipal();
        private HashMap<String, Actor> milista;


        //Constructora
        public ListaActoresPrincipal(){
                milista = new HashMap<String, Actor>();
        }


        //Getters y Setters
        public static ListaActoresPrincipal getListaActoresPrincipal(){
                return milistadeactores;
        }

        public HashMap<String,Actor> getMilista(){
                return this.milista;
        }

        public Iterator<String> getIterador(){
                return this.getMilista().keySet().iterator();
        }

        public int getTamano(){
                return this.getMilista().size();
        }


        //Otros Metodos
                public boolean esta(Actor pActor){
                        return this.getMilista().containsKey(pActor.getNombre());
                }
```

```
        public void anadirActor(Actor pActor){
                if(esta(pActor)){
                        System.out.println("El actor ya se encuentra en la lista");
                }else{
                        getMilista().put(pActor.getNombre(), pActor);
                }
        }

public void eliminarActor(Actor pActor){
        try{
                if(esta(pActor)){
                        getMilista().remove(pActor);
                }else{
                        System.out.println("El actor no se encuentra en la lista");
                }
        }catch(NullPointerException e){
                System.out.println("El actor que desea eliminar no existe");
        }
}

public Actor buscarActor(Actor pActor){
        try{
                if(esta(pActor)){
                        return this.getMilista().get(pActor.getNombre());
                }else{
                        return null;
                }
        }catch(NullPointerException e){
                System.out.println("El actor que intentas buscar no existe");
                return null;
        }
}

public Actor buscarActorNombre(String pActor){
        //Devuelve el Actor si está y sino devuelve null
        return this.getMilista().get(pActor);
}

public void resetear(){
        getListaActoresPrincipal().getMilista().clear();
}

public void imprimir(){
        Iterator<String> it = getListaActoresPrincipal().getIterador();
        while(it.hasNext()){
                getListaActoresPrincipal().getMilista().get(it.next()).imprimirInformacion();
                System.out.println("\n\n");  //Imprime dos lineas vacías.
        }
}

public void ordenarLista(){
        Actor[] milistaordenada = this.convertirHashArray();
        ordenacionPorBurbuja(milistaordenada);
        //JOptionPane.showMessageDialog(null, "El resultado se muestra por consola");
        for(int i = 0; i<milistaordenada.length; i++)
                System.out.println(i+": "+milistaordenada[i].getNombre());
}

public Actor[] convertirHashArray(){
        Actor[] miArrayDeActores = new Actor[this.getMilista().size()];
        int i = 0;
        Iterator<String> itr = this.getIterador();
        while (itr.hasNext()){
                miArrayDeActores[i] = this.getMilista().get(itr.next());
                i++;
        }
        return miArrayDeActores;
}

public void ordenacionPorBurbuja(Actor[] tabla) {
        int out, in;
        for (out = tabla.length - 1; out > 0; out--)
                for (in = 0; in < out; in++)
                        if ( tabla[in].compareTo(tabla[in + 1]) > 0 )
                                swap(tabla, in, in + 1);
}
```

```
        private void swap(Actor[] tabla, int a, int b) {
                Actor aux = tabla[a];
                tabla[a] = tabla[b];
                tabla[b] = aux;
        }
```

5.1.7 }Clase ListaPeliculasPrincipal

```
package lab3;

import java.util.HashMap;
import java.util.Iterator;

public class ListaPeliculasPrincipal {
        //Atributos
        private static ListaPeliculasPrincipal milistaprincipaldepeliculas = new ListaPeliculasPrincipal();
        private HashMap<String, Pelicula> lista;


        //Constructora
        public ListaPeliculasPrincipal(){
                this.lista = new HashMap<String, Pelicula>();
        }


        //Getters y Setters
        public static ListaPeliculasPrincipal getListaPeliculasPrincipal(){
                return milistaprincipaldepeliculas;
        }


        private HashMap<String, Pelicula> getLista(){
                return this.lista;
        }

        public Iterator<String> getIterador(){
                return this.getLista().keySet().iterator();  //Esto crea un iterador de keys o llaves

//en nuestro caso titulos de peliculas.
        }


        //Otros Metodos
        public boolean estaPelicula(Pelicula pPelicula){
                return this.getLista().containsKey(pPelicula.getTitulo());
        }

        public void anadirPelicula(Pelicula pPelicula){
                if((pPelicula!=null)&&(!estaPelicula(pPelicula))){
                        this.getLista().put(pPelicula.getTitulo(), pPelicula);
                }else{
                        System.out.println("la pelicula introducida no es valida.");
                }
        }

        public void eliminarPelicula(Pelicula pPelicula){
                if((pPelicula!=null)&&(estaPelicula(pPelicula))){
                        this.getLista().remove(pPelicula);
                }else{
                        System.out.println("La pelicula no se encuentra en la lista.");
                }
        }

        public void resetearLista(){
                this.getLista().clear();
        }

        public Pelicula buscarPeliNombre(String pTitulo){
                //Devuelve la Película si está y sino devuelve null
                return this.getLista().get(pTitulo);
        }
}
```

5.1.8 Clase Cola

```
package lab3;

public class Cola<T> {

	private T[] cola;
	private int max;
	private int numeroelementos;
	private int primero;
	private int ultimo;

	public Cola(int pMax){
		cola = (T[]) new Object[pMax];
		max = pMax;
		primero = 0;
		numeroelementos=0;
		ultimo = -1;
	}

	public int getTamano(){
		return this.numeroelementos;
	}

	public void anadir(T pAnade){
		if(numeroelementos<max){
			numeroelementos++;
			if(ultimo==max-1){
				System.out.println("true");
				ultimo=-1;
			}
			ultimo++;
			cola[ultimo]=pAnade;
		}
	}

	public T sacarPrimerElemento(){
		T elem = null;
		if(numeroelementos!=0){
			elem=cola[primero];
			primero++;
			if(primero==max){
				primero=0;
			}
			numeroelementos--;
		}
		return elem;
	}

	public void imprimirCola(){
		int cont = 0;
		int aux = primero;
		while(cont<numeroelementos){
			System.out.println(cola[aux]);
			aux++;
			cont++;
			if(aux==max){
				aux=0;
			}
		}
	}

}
```

5.1.9 Clase PruebasCola

```
package lab3;

public class PruebasCola {

	public static void main(String[] args){
		Cola<Integer> c = new Cola<Integer>(3);
		System.out.println("Añado el 3");
		c.anadir(3);
		System.out.println("El tamaño de la cola tiene que ser 1 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Añado el 6");
		c.anadir(6);
		System.out.println("El tamaño de la cola tiene que ser 2 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Añado el 23");
		c.anadir(23);
		System.out.println("El tamaño de la cola tiene que ser 3 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Añado el 567");
		c.anadir(567);
		System.out.println("El tamaño de la cola tiene que ser 3 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Saco el primero");
		c.sacarPrimerElemento();
		System.out.println("El tamaño de la cola tiene que ser 2 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Añado el 400");
		c.anadir(400);
		System.out.println("El tamaño de la cola tiene que ser 3 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Saco el primero");
		c.sacarPrimerElemento();
		System.out.println("El tamaño de la cola tiene que ser 2 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Saco el primero");
		c.sacarPrimerElemento();
		System.out.println("El tamaño de la cola tiene que ser 1 y es: "+c.getTamano());
		c.imprimirCola();

		System.out.println("Añado el 4142");
		c.anadir(4142);
		System.out.println("El tamaño de la cola tiene que ser 2 y es: "+c.getTamano());
		c.imprimirCola();
	}

}
```

6 Conclusiones

En este tercer laboratorio hemos continuado trabajando sobre las listas y la estructura Hash Map, además hemos utilizado la estructura de datos Cola. El problema que se nos pidió resolver lo vimos bastante claro desde un principio, sin embargo, dar con la solución adecuada nos ha requerido tiempo y nos ha hecho pensar. Hemos recorrido bastante camino hasta dar con la mejor solución.